باسمه تعالی

در این نوشتار، مباحثی در مورد نقش شبه‌سازی و محاسبات رایانه‌ای را به طور کلی و خصوصاً در فیزیک و همچنین مهمترین نکات مربوط به نوشتن یک برنامه رایانه‌ای را خصوصاً به زبان فرترن ارایه می‌گردد.

❖ نقش شبیه‌سازی

ایجاد یک ساختار و فناوری جدید نیازمند مطالعات عمیق تئوری و آزمایش‌های تجربی می‌باشد. برای نیل به این مهم بایستی از تمام امکانات به صورت بهینه بهره گرفت. گاهی ساختار پیچیده یک سیستم، هزینه مالی و زمانی زیادی را برای تکمیل و بهره برداری از آن در مقیاس تجاری به خود اختصاص می‌دهد، بنابراین یافتن روش‌هایی جهت بهینه کردن هزینه‌ها نه تنها لازم بلکه ضروری است.

پیشرفت‌های قابل توجه اخیر در حوزه فناوری رایانه‌ها از یک سو و از سویی دیگر بلوغ جمعی علوم پایه و فنی- مهندسی به همراه علوم زیست- محیطی شرایطی را فراهم کرده تا رهیافتهایی نوین جهت حل مسایل پیچیده و همچنین پرداختن به حوزه‌هایی فراتر از پژوهش‌های سنتی، مورد کنکاش و توجه واقع گردد. با توجه به چارچوب مناسبی که از این منظر فراهم آمده‌است، ارایه راهکارهایی جدید برای تقاضاهای درونی و برونی هر شاخه از علم، طی یک روش بهینه و منطقی در بسیاری از موارد همراهی و هم‌افزایی چندین شاخه از علوم را مبتنی بر روشهای محاسباتی نوین را طلب می‌کند. به بیانی دیگر پیچیدگی‌های علوم نوین که از یک سو خود معلول پیشرفتهای چشم‌گیر علوم پایه می‌باشد به همراه درخواست جامعه بشری، عامل اصلی بنای علوم بین‌رشته‌ای و ابزارهای جدید برای واکاوی و انجام محاسبات می‌باشند.

با رشد صنعت رایانه و حضور گسترده رایانه‌هایی با سرعت پردازش اطلاعات بالا روشهای محاسباتی و شبیه‌سازی در هر شاخه‌ای از علم رشد چشم‌گیری داشته و امروزه شاید یکی از بهترین و مقرون به صرفه‌ترین راههای مطالعه اولیه سیستم‌ها در شرایط متفاوت حاکم بر آنها می‌باشد. روشهای محاسباتی و شبیه‌سازی از مقیاس نانو گرفته تا مقیاسهای کیهانی و انرژی‌های بالا نقش ارزنده‌ای را ایفا کرده و می‌کند. در واقع بدون بهره‌گیری از روشهاس محاسباتی بدون گزافه‌گویی می‌توان گفت زمانی در حدود صدها میلیون سال مورد نیاز است تا ماهیت ساختارهای نانومقیاس و بزرگ مقیاس، خواص و پایداری آنها تعیین شود. با توجه به دقت اطلاعات دریافتی از روشهای محاسباتی این روشها امروزه به عنوان یکی از مهمترین و کارآمدترین ابزار نسبتا ارزان قیمت و سریع برای بررسی تمام خواص سیستم‌های مطرح می‌شود. با بهره‌گیری از این روشها می‌توان بدون صرف هزینه اضافی در آزمایشگاهها خواص سیستم‌ها را تا حد قابل قبولی پیشگویی کرد. البته این نکته قابل بررسی است که در برخی موارد به دلیل بالا بودن حجم محاسبات بایستی از ابر رایانه بهره گرفت ولی آنچه امروزه به اثبات رسیده است بیشتر سیستمها با کمک رایانه‌های شخصی یا تعدادی رایانه موازی شده قابل بررسی می‌باشند.

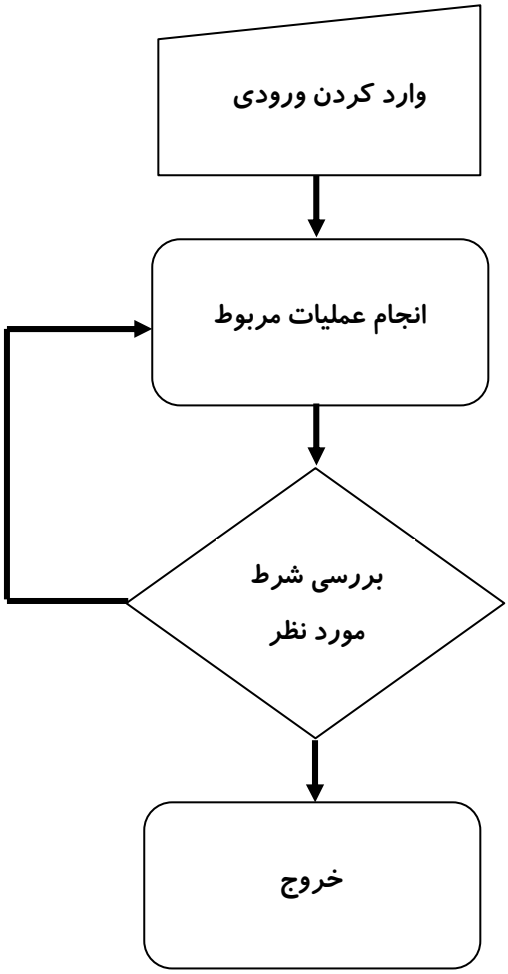
در بحث تئوری، مدل‌سازی و شبیه‌سازی مهم‌ترین پیشرفت کاربردی، همراه کردن بسیاری از رایانه‌های پرقدرت و نرم‌افزارهای پیشرفته متناظر با تئوری‌های جدید می‌باشد.

به عنوان جمع‌بندی می‌توان گفت که تئوری مدلسازی و شبیه‌سازی و بکاربردن الگوریتم‌های محاسباتی نقش مهمی برای نیل به اهداف زیر بازی می‌کند.

1. **کاهش زمان مورد نیاز برای طراحی سیستم‌های جدید**
2. **افزایش اطمینان در قدرت پیش‌گویی از عملکرد سیستم‌های جدید**
3. **فراهم آوردن شرایط بررسی تطابق مشاهدات و تئوری‌ها**

❖ نکات و توصیه‌های ضروری برای برنامه نویسی

۱) قبل از نوشتن برنامه، ابتدا بر روی کاغذ به صورت طراحی، الگوریتم برنامه مورد نظر را پیاده نمایید. برای مثال الگوریتم نمایشی برنامه‌ای که با گرفتن یک ورودی، عملیات خاصی را روی آن انجام می‌دهد و با توجه به یک شرط برنامه را به پایان می‌دهد به صورت زیر است:



۲) انتخاب زبان برنامه‌نویسی در حالت کلی اختیاری است اما از آنجا که دو زبان Fortran و C در شبیه سازی‌های زیادی به کار می‌رود و تقریباً هیچ برنامه مهمی را پیدا نمی‌کنید که به یکی از این دو زبان نوشته نشده‌باشند. بنابراین توصیه می‌شود که به یکی از این دو برنامه کد بنویسید.

۳) برای خوانا شدن برنامه، برای هر قسمت یک توضیح بنویسید. در فرترن اگر "!" را ابتدای هر خط به کار برید اجرا کننده هر چیزی بعد از آن را صرف نظر می‌کند.

۴) فایل برنامه در fortran90 (file.f90) دارای فرمت آزاد است ولی در fortran77 (file.f) باید از ستون ۷ شروع به نوشتن نمایید.

۵) از آنجا که در فرترن هیچ محدودیتی برای نامگذاری نیست بنابراین بهتر است از نامهای به اندازه کافی واضح برای متغیرها استفاده کنید و هرگز خساست برای نام نهادن به کار نبرید.

۶) برای اجتناب از خطا در مقدارگیریِ کمیت‌ها حتماً از دستور implicit none استفاده نمایید.

۷) برای استفاده بهینه از حافظه رایانه، از دستورهای ALLOCATE و DEALLOCATE برای آرایه‌ها استفاده کنید. همچنین از توابع داخلی آن تا آنجا که امکان دارد بهره بگیرید. مثلاً $C_{ij} = \sum_k A_{ik} B_{kj} \rightarrow C = matmul(A, B)$

۸) یک برنامه که اصطلاحاً Procedure oriented programming(POP) نامیده می شود، شامل یک مجموعه دستورات و توابع است که پشت سرهم در برنامه نوشته شده و توسط اجرا کننده خوانده شده و اجرا می‌شود. حتی در برنامه‌های پیچیده‌تر این وضعیت شامل توابع و زیرروالها (subroutine) می‌باشد. و البته بسیاری از متغیرها به طور گسترده (global) قابل دستیابی و تغییر هستند. اما در یک برنامه شئ‌گرا (Object oriented programming (OOP)) چه اتفاقی می‌افتد؟ ابتدا مفهوم شئ را مطرح می‌کنم. یک شئ دارای دو خاصیت عمده یعنی وضعیت و رفتار است. دقیقاً معادل تعریف شئ در

دنیای واقعی. برای مثال چراغ مطالعه یک شئ است که دارای دو حالت روشن و خاموش است و رفتار مترتب به آن روشن کردن و خاموش کردن، است. در یک برنامه رایانه‌ای یک شئ دارای حالت است که توسط متغیرها توصیف می‌شود. و رفتار آن نیز در چارچوب روشها (توابع و غیره) خلاصه می‌شود. بنابراین می‌توان گفت که

- در POP اهمیت به مجموعه‌ای از کارهایی که باید انجام شود داده می‌شود در صورتی که در OOP این اهمیت به داده‌ها منتسب می‌شود.
- در POP بخشهای مختلف به توابع تقسیم‌بندی می‌شوند در صورتی که در OOP بخشهای مختلف به اشیاء تقسیم‌بندی می‌شوند.
- در POP عموماً داده‌های توابع به صورت گسترده قابل دسترس هستند در حالی که در OOP داده‌ها عموماً خصوصی هستند.
- POP یک رهیافت بالا به پایین است در حالی که در OOP قضیه عکس است.
- در POP گزینه‌های مختلف برای دسترسی وجود ندارد در حالی که در OOP گزینه‌هایی مثل public، private و protected وجود دارد.

۹) سعی کنید تا آنجا که امکان دارد با کاربر رابطه دوستانه برقرار کنید به بیانی دیگر تا آنجا که امکان دارد از واسط‌های گرافیکی بهره بگیرید.

مثال: نمونه زیر شامل مهمترین دستورات یک برنامه فرترنی POP می‌باشد:

متغیرها معرفی می گردند (global)

```
module parameters
use numerical_libraries
implicit none
real(8),parameter::num=150
character(256)::numstring
real(8) x,yy,y,q1(-100:100),q2(0:100),q3(-100:100,0:10)
Real*8, ALLOCATABLE :: z1(:),P12(:,:),z2(:)
INTEGER    IR, IS, J
COMPLEX(8)    C, CEXP, CMPLX, COEF(num,num),coef_cmb(num,num)
end module parameters
!************************Main program **************
main_program
use parameters
implicit none
INTEGER, PARAMETER:: double=SELECTED_REAL_KIND(15,307)
real(double) CHSQ,DF1,t2,chi_sq_GSN_GN, &

&     P_value_t_GSN_GN(0:1000),t_GS_G(0:10000)
integer LDB,num11
open(10,file='input_file.txt')    !*********** file explanation
pi=4*atan(1.0)
!*********** read data from input file**************
do i =1,10000000
      read(10,*,end=9)dummy
      counter=counter+1
enddo
9     rewind(10)
allocate (z1(counter), z2(counter), P12(counter,conter))
do i=1,counter
      read(10,*) z1(i), z2(i)
enddo
call initial_condition    !****** call a typical subroutine
call random_seed
call noise
call gaussian_map
call print_result
!***********LOOP of RK4
a_final=10
a=0
```

```
      step=1
      do while(a.le.a_final)
            i=i+1
            a=a+step
            f1(i)=x
            f2(i)=y
            call RK4
            write(30,*)a,f1(i)   !******* make fort.30 as an output file
      enddo
      deallocate (z1,z2,P12)
      end maine_program
!************ subroutines
subroutine initial_condition
use parameters
implicit none
im=(0,1)
lda=num
nca=num
nra=num
end subroutine initial_condition
!******************************
subroutine noise
use parameters
implicit none
call random_seed
do i=1,num**2
call random_number(x)
call random_number (y)
z1(i)=sqrt(-2*log(x))*cos(2*pi*y)
z2(i)=sqrt(-2*log(x))*sin(2*pi*y)
enddo
End subroutine noise
!*****************
subroutine gaussian_map
use parameters
implicit none
! *************************** Backward FFT of MAP ***************
CALL DFFT2B (NRA, NCA, COEF_cmb, LDCOEF, temp1, LDA)
end subroutine gaussian_map
subroutine RK4
use parameters
k1=step*fun1(a,y,x)
yy=y+(1.0/6.0)*(k1**2)
y=yy
xx=x+(1.0/6.0)*(k1)
x=xx
end subroutine RK4

subroutine print_results
use parameters
implicit none
write(numstr,'(i1)') n_run
numstring='GSN.'//numstr
open(70,file=numstring)
do j=1,num
write(70,*)i,j,temp_f(i,j)
enddo
end subroutine print_results
function fun1(a11,y11,x11)
use parameters
real(8):: a11,y11,x11
fun1=y11
return
end
```

****

خیلی از مواقع می‌توان از یک فایل استفاده کرد و در آن تمام کارهایی که می‌خواهیم به صورت پشت سر هم انجام دهیم را در آن بنویسیم و بعد با اجرای آن به یک باره تمام فرمانها اجرا می‌شوند. برای این کار یک فایل با پسوند .csh یا .sh بسته به اینکه cshell یا bash انتخاب شده باشد. در زیر یک نمونه cshell script آمده است.

```
A sample for cshell script
#!bin/csh/
mkdir /Users/sadeghmovahed/Desktop/GN
set i=0
while ($i < 100)
echo $i
@    i = ($i + 1)
cp GN.${i} test
g++ /Users/sadeghmovahed/Desktop/all/converter.cpp
/a.out
Ifort /Users/sadeghmovahed/Desktop/all/fortran_file1.f90 -o fortran1_out.exe
/fortran 1_out.exe
Gfortran /Users/sadeghmovahed/Desktop/all/fortran_file2.f -o fortran2_out.exe
/fortran 2_out.exe
mv output.txt /Users/sadeghmovahed/Desktop/GN/GN${i}.txt
end
```

پس از آن باید این فایل را اجرایی کرد. برای این کار ابتدا یک فولدر درست می‌کنیم بعد فایل خود را در آن قرار می‌دهیم و مراحل زیر انجام می‌شود:

```
$ chmod u+x sample.csh
$ ./sample.csh
```

مثالی در مورد یک برنامه که به صورت شئ گرا OOP نوشته شده‌است (http://fortranwiki.org):

```
module class_Circle
      implicit none
      private
      public :: Circle, circle_area, circle_print
      real :: pi = 3.1415926535897931d0 ! Class-wide private constant
      type Circle
       real :: radius
       end type Circle
      contains
      function circle_area(this) result(area)
      type(Circle), intent(in) :: this
      real :: area
      area = pi * this%radius**2
       end function circle_area
      subroutine circle_print(this)
       type(Circle), intent(in) :: this
       real :: area
       area = circle_area(this)  ! Call the circle_area function
       print *, 'Circle: r = ', this%radius, ' area = ', area
       end subroutine circle_print
end module class_Circle
program circle_test
 use class_Circle
 implicit none
 type(Circle) :: c    ! Declare a variable of type Circle(برای شئ یک حالت نسبت داده‌است)
 c = Circle(1.5)       ! Use the implicit constructor, radius = 1.5.
 call circle_print(c)  ! Call a class subroutine
end program circle_test
```

سیدمحمدصادق موحد

www.smovahed.ir

Unix/Linux Command Reference

FOSSwire.com

File Commands

ls – directory listing
ls -al – formatted listing with hidden files
cd *dir* - change directory to *dir*
cd – change to home
pwd – show current directory
mkdir *dir* – create a directory *dir*
rm *file* – delete *file*
rm -r *dir* – delete directory *dir*
rm -f *file* – force remove *file*
rm -rf *dir* – force remove directory *dir* *
cp *file1 file2* – copy *file1* to *file2*
cp -r *dir1 dir2* – copy *dir1* to *dir2*; create *dir2* if it doesn't exist
mv *file1 file2* – rename or move *file1* to *file2*
if *file2* is an existing directory, moves *file1* into directory *file2*
ln -s *file link* – create symbolic link *link* to *file*
touch *file* – create or update *file*
cat > *file* – places standard input into *file*
more *file* – output the contents of *file*
head *file* – output the first 10 lines of *file*
tail *file* – output the last 10 lines of *file*
tail -f *file* – output the contents of *file* as it grows, starting with the last 10 lines

Process Management

ps – display your currently active processes
top – display all running processes
kill *pid* – kill process id *pid*
killall *proc* – kill all processes named *proc* *
bg – lists stopped or background jobs; resume a stopped job in the background
fg – brings the most recent job to foreground
fg *n* – brings job *n* to the foreground

File Permissions

chmod *octal file* – change the permissions of *file* to *octal*, which can be found separately for user, group, and world by adding:

- 4 – read (r)
- 2 – write (w)
- 1 – execute (x)

Examples:
chmod 777 – read, write, execute for all
chmod 755 – rwx for owner, rx for group and world
For more options, see **man chmod**.

SSH

ssh *user@host* – connect to *host* as *user*
ssh -p *port user@host* – connect to *host* on port *port* as *user*
ssh-copy-id *user@host* – add your key to *host* for *user* to enable a keyed or passwordless login

Searching

grep *pattern files* – search for *pattern* in *files*
grep -r *pattern dir* – search recursively for *pattern* in *dir*
command* | grep *pattern – search for *pattern* in the output of *command*
locate *file* – find all instances of *file*

System Info

date – show the current date and time
cal – show this month's calendar
uptime – show current uptime
w – display who is online
whoami – who you are logged in as
finger *user* – display information about *user*
uname -a – show kernel information
cat /proc/cpuinfo – cpu information
cat /proc/meminfo – memory information
man *command* – show the manual for *command*
df – show disk usage
du – show directory space usage
free – show memory and swap usage
whereis *app* – show possible locations of *app*
which *app* – show which *app* will be run by default

Compression

tar cf *file.tar files* – create a tar named *file.tar* containing *files*
tar xf *file.tar* – extract the files from *file.tar*
tar czf *file.tar.gz files* – create a tar with Gzip compression
tar xzf *file.tar.gz* – extract a tar using Gzip
tar cjf *file.tar.bz2* – create a tar with Bzip2 compression
tar xjf *file.tar.bz2* – extract a tar using Bzip2
gzip *file* – compresses *file* and renames it to *file.gz*
gzip -d *file.gz* – decompresses *file.gz* back to *file*

Network

ping *host* – ping *host* and output results
whois *domain* – get whois information for *domain*
dig *domain* – get DNS information for *domain*
dig -x *host* – reverse lookup *host*
wget *file* – download *file*
wget -c *file* – continue a stopped download

Installation

Install from source:
./configure
make
make install
dpkg -i *pkg.deb* – install a package (Debian)
rpm -Uvh *pkg.rpm* – install a package (RPM)

Shortcuts

Ctrl+C – halts the current command
Ctrl+Z – stops the current command, resume with **fg** in the foreground or **bg** in the background
Ctrl+D – log out of current session, similar to **exit**
Ctrl+W – erases one word in the current line
Ctrl+U – erases the whole line
Ctrl+R – type to bring up a recent command
!! - repeats the last command
exit – log out of current session

* use with extreme caution.

How to Use Emacs

For a complete guide to `emacs`, check out the GNU Emacs Online Manual, or see *Learning GNU Emacs* by Debra Cameron and Bill Rosenblatt (O'Reilly & Associates, Sebastopol, CA, 1991, ISBN 0-937175-84-6).

Emacs For the Dummies

To run emacs, just type

```
emacs filename
```

where *filename* is the file you want to edit.

On PowerPC desktop or any workstations with X windows, you may want to put the emacs session into background so that you can still use the current xterm window, just type

```
emacs filename &
```

Once you are in emacs, the top menu bars offer the four sub-menus:

```
Buffer   File   Edit    Help
```

The sub-menu `Buffer` allows you to switch between different files that you are editing. The sub-menu `File` contains commands on how to open another file, save files, exit emacs, etc. For example, to open a new file, just click the `File` button and the `Open File...` option, the cursor will then jump to the minibuffer at the bottom of the screen; you can type in the file name, the one which you want to open, and type the "return" key.

Alternatively (preferred by most people), you can use the key bindings to do most of these and more.

```
Use the arrow keys to move the cursor

C-x C-f         open a new file
C-x C-s         save the current file
C-x C-c         exit the emacs (but save files first)
```

Here, the prefix `C-` refers to the CONTROL key, the prefix `ESC-` refers to the ESCAPE pkey. For example, `C-x` means to simultaneously press the CONTROL key and the "x" key.

Key Bindings

In the following, the prefix C- refers to the CONTROL key, the prefix ESC- refers to the ESCAPE key. For example, C-n means to simultaneously press the CONTROL key and the key "n".

Lines

```
C-a             go to the beginning-of-line
C-e             go to the end-of-line
C-n             go to next-line
C-p             go to previous-line
C-k             kill the current line
C-o             open-line

The following two bindings are CS210 specific:

  C-x C-g         go to a specific line numbered x
  C-x C-w         show (in the minibuffer ) the current line number
```

Words

```
ESC f           forward-word
ESC b           backward-word
ESC d           kill-word
ESC DEL         backward-kill-word
```

Characters

```
C-f             forward-char
C-b             backward-char
C-d             delete-char
DEL             delete-backward-char
C-q             quoted-insert
C-t             transpose-chars
```

Regions

```
C-space         set a region mark
C-w             kill-region (between cursor and mark)
ESC-w           memorize the contents of the region (without kill)
C-y             yank (i.e., insert text last killed or memorize)
```

Screen control

```
C-l             recenter
C-v             scroll-up (forward)
ESC-v           scroll-down (backward)
ESC <           beginning-of-buffer
ESC >           end-of-buffer
```

Search

```
C-s             isearch-forward
C-r             isearch-backward
```

Files

```
C-x C-f         find-file
C-x C-r         find-file-read-only
C-x C-s         save-current-file
```

Windows

```
C-x 1           delete-other-windows
C-x 2           split-window-vertically
C-x 4 f         find-file-other-window
C-x o           other-window
```

Command execution

```
ESC !           shell-command
ESC x compile   compile ("make -k" is default)
C-x `           next-error
                (used after "compile" to find/edit errors)
```

Miscellaneous

```
C-x C-c         save-buffers-kill-emacs
C-u             universal-argument
C-x C-z         suspend-emacs
                (resume by typing "fg" to unix)
```

Help!

```
C-g             keyboard-quit
C-h             help-command
C-h t           help-with-tutorial
C-h b           describe-bindings
                (complete list of emacs commands)
```

Commands for Compiling

`ESC-x compile`
runs the compiler, linker, etc.

If this is the first time you have issued this command since entering emacs, the minibuffer at the bottom of the screen appears filled with `make -k`. If you're not using `make -k` erase the minibuffer line (e.g., using `DEL`) and type in the compiler command of your choice, e.g., `gccx hello.c`. This command is remembered for subsequent executions of `ESC-x compile`. When you type `RETURN`, if there are unsaved buffers, you will be given the opportunity to save each one. The screen then splits into two windows, and the output from the compilation command appears in one of the two windows. If there are parse errors, use the following command.

`C-x ``
finds the locations of errors. Each time this command is given after a compilation that found errors, another line of parse errors is located. The compilation window is scrolled up, so that the topmost line displays the a new parse error. The other window changes buffers, if necessary, and displays the source line associated with the error. Note that if your program consists of several files, this command locates the file and loads it into the buffer. The cursor is placed at the line containing

the error. You may edit the file to correct the source of the error and repeat the command again to find additional errors. When you have done the most you can with this batch of parse errors, give the `ESC-x compile` command again.

Commands for Debugging

`ESC-x gdb`
: runs gdb, the GNU interactive debugger.

The minibuffer at the bottom of the screen prompts you for the name of your executable file. Unless you compiled with the `-o` option to name the output file, the name of your executable file is `a.out`. When you type in the file name followed by `RETURN`, the screen splits into two windows (or remain split if it is split already). One window is used for interactive input and output to gdb. The other will eventually display your program files for you to examine and edit. Sometimes the screen doesn't divide immediately after `ESC-x gdb`, but `gdb` takes over the whole window where it was executed from; the screen divides the first time you run the program and it stops because of a breakpoint or an error caught by the debugger. So, if the window doesn't split and you want to follow the behaviour of the running program, just type `break main` before you run it the first time. When execution reaches `main`, the window splits as described above, an arrow points to the current position in the code, which is the first line of main.

The cursor initially is placed after the `gdb` prompt `(gdb)`. Whenever you want to issue a command to gdb, position the cursor at the end of the buffer, i.e., after the `(gdb)`, and type the command as usual. The command `ESC->` gets you to the end of the buffer. To examine previous input or output to `gdb`, use the usual emacs commands to move around the buffer.

Whenever your program, which was running under `gdb`, stops because of a breakpoint an interrupt, etc., the source code associated with the current locus of execution is displayed automatically in the other window. A marker, `=>`, points to the specific line. If you use the `frame` command to change frames, the source for the new frame is displayed and the marker is placed accordingly.

When you find an error, you may change the source code and save the file. However, before recompiling, give `gdb` the command `kill` to cancel your running program. Otherwise, when the compiler runs the linker to link your program you'll get the error `text file busy' and a new executable file will not be written.

After recompiling a program, you should reload the symbol table and the executable, otherwise you'll be running the previous program. To do so, execute

```
(gdb) exec-file program-name
(gdb) symbol-file program-name
```

The `symbol-file` command will request confirmation before reloading the symbol table; just answer `yes`.

`ESC-x gdb-break`
: sets a `gdb` breakpoint at the source line on which the cursor appears.

Commands for Controlling Windows

Windows in emacs usually refer to subdivisions.

`C-x 1`
: reformats screen into one window, retaining only the window in which the cursor appears.

`C-x 4 f`
: finds a file and displays it in the other window (the window in which the cursor does not appear). If the screen has only one window, split it into two. The `C-x 4f` command prompts for the file name.

`C-x o`
: moves the cursor to another window.

Windows in the usual sense are called **frames** in emacs. You can use frames only if you are using PowerPC desktops or any workstations with X window systems. Frames give you true separate emacs windows at the same time. Clearly, you can not use frames if you are using a text-based terminal (e.g., via remote telnet or remote login).

`C-x 5 2`
: Creates a new frame.

`C-x 5 f`
: Finds a file and displays it in the other frame (the frame in which the cursor does not appear).

`C-x 5 0`
: Deletes the selected frame. This is not allowed if there is only one frame.

---

rm -r delete folder
cp -r copy folder
mkdir -p make a folder without error if existing the same folder

change the permission of folder
chmod -R 777 name of file of folder

to change the mode for executive file
chmod u+x sample.sh

to run bash file
./sample.sh

to run a script for gnuplot
sample.gnu
gnuplot sample.gnu

for getting a file from linux when I am in mac or other machine:

scp movahed@194.225.68.237:/home/movahed/Desktop/filename "target address e.g. /Users/sadeghmovaged/Desktop"

for sending a file form my machine to a server

scp /Users/sadeghmovahed/Desktop/collaboration/Hurst_surface/Movahed-tm40.zip movahed@194.225.68.253:/home/movahed/Desktop/

connecting to a server
ssh -x movahed@194.******

running a program while the terminal to be closed
nohup ./fort.f90

how to connect from linux to windows

smb://Administrator@192.168.80.52

n_run is a number using following commend in fortran it is moved to a character and it size is “i1”

write(numstr,'(i1)') n_run

numstring='GSN.'//numstr

open(70,file=numstring)

or (without any limitation in size of initial value we select as large as possible type and by trim we discard the tail of this character as follows (e.g n_run=10 while i10 mean a place for ********** (no. of stars is 10) so with out using trim we find

GSN. 10 instead of having GSN.10)

write(numstr,'(i10)') n_run

numstring='GSN.'//trim(adjustl(numstr))

getting and installing application

sudo apt-get install “name of application”

for marging two pdf file fist of all we should alias the command as follows

in terminal write

```
cd /usr/local/bin
su movahed ln "/System/Library/Automator/Combine PDF Pages.action/Contents/
Resources/join.py" PDFconcat
```

hereafter in terminal for merging two file wite

PDFconcat -o /Users/movahed/Desktop/final.pdf first_file.pdf second_file.pdf

```
#! /bin/bash

#Reading of contries list
i=0
for name in `cat list_arrange` ; do
  let "i = $i+1"
  c[i]=$name
#echo $name
done

let "num = $i"

#calculating
add=$(pwd)
ifort DCCA_new_new.f90 -o dcca
ifort DCCA_new_new100.f90 -o dcca100
ifort DCCA_first100.f90 -o dccafirst100
ifort DCCA_second100.f90 -o dccasecond100
#ifort DCCA_first300.f90 -o dccafirst300
#ifort DCCA_second300.f90 -o dccasecond300

ifort DFA_second300.f90 -o dfasecond300
ifort DFA_first300.f90 -o dfafirst300


for ((i=1; i<=$num; i++)); do
let "k = $i+1"
for ((j=$k; j<=$num; j++)); do
#echo "${c[i]}"

mkdir -p $add/${c[i]}_${c[j]}
cp  $i.txt first.txt
cp  $j.txt second.txt

y=0
g=0
#***************************
if [  "${c[i]}" = "EGYPT" ];then
if [  "${c[j]}" = "PAKISTAN" ];then
./dcca100
y=1
fi
fi

if [  "${c[j]}" = "EGYPT" ];then
if [  "${c[i]}" = "PAKISTAN" ];then
./dcca100
y=1
fi
fi
#*******************************
```

```
#-or "c[i]" = "PAKESTAN" -or "c[j]"="PAKESTAN"
#if [ "c[i]" = "EYGEPT" -or "c[j]"="EYGEPT"];  then
if [  "$y" = "0" ];then
o=0
if [  "${c[i]}" = "EGYPT" ];then
echo ${c[i]}
./dccafirst100
./dfasecond300
o=1
fi
if [  "${c[j]}" = "EGYPT" ];then
./dccasecond100
./dfafirst300
echo ${c[j]}
o=1
fi
if [  "${c[i]}" = "PAKISTAN" ];then
./dccafirst100
./dfasecond300
o=1
fi
if [  "${c[j]}" = "PAKISTAN" ];then
./dccasecond100
./dfafirst300
o=1
fi
# cp input_win100 max_win_input
if [  "$o" = "0" ];then
#cp input_win300 max_win_input
./dcca
fi
fi
rm -f first.txt
rm -f second.txt

mv hq_first $add/${c[i]}_${c[j]}/hq_$i
mv hq_second $add/${c[i]}_${c[j]}/hq_$j
mv hq_DCCA $add/${c[i]}_${c[j]}/hq_${i}_$j

mv tauq_first $add/${c[i]}_${c[j]}/tauq_$i
mv tauq_second $add/${c[i]}_${c[j]}/tauq_$j
mv tauq_DCCA $add/${c[i]}_${c[j]}/tauq_${i}_$j

mv Dq_first $add/${c[i]}_${c[j]}/Dq_$i
mv Dq_second $add/${c[i]}_${c[j]}/Dq_$j
mv Dq_DCCA $add/${c[i]}_${c[j]}/Dq_${i}_$j

mv falpha_first $add/${c[i]}_${c[j]}/falpha_$i
mv falpha_second $add/${c[i]}_${c[j]}/falpha_$j
mv falpha_DCCA $add/${c[i]}_${c[j]}/falpha_${i}_$j

mv falpha_first_error $add/${c[i]}_${c[j]}/falpha_error_$i
mv falpha_second_error $add/${c[i]}_${c[j]}/falpha_error_$j
```

```
mv falpha_DCCA_error $add/${c[i]}_${c[j]}/falpha_error_${i}_$j

mv delta_falpha_first $add/${c[i]}_${c[j]}/delta_falpha_$i
mv delta_falpha_second $add/${c[i]}_${c[j]}/delta_falpha_$j
mv delta_falpha_DCCA $add/${c[i]}_${c[j]}/delta_falpha_${i}_$j



mv fort.43333 $add/${c[i]}_${c[j]}/delta_falpha_error_${i}_$j
mv fort.23333 $add/${c[i]}_${c[j]}/delta_falpha_error_$i
mv fort.33333 $add/${c[i]}_${c[j]}/delta_falpha_error_$j

mv fort.114 $add/${c[i]}_${c[j]}/Hurst_$i
mv fort.1144 $add/${c[i]}_${c[j]}/Hurst_error_$i

mv fort.115 $add/${c[i]}_${c[j]}/Hurst_$j
mv fort.1155 $add/${c[i]}_${c[j]}/Hurst_error_$j


mv fort.116 $add/${c[i]}_${c[j]}/Hurst_${i}_$j
mv fort.1166 $add/${c[i]}_${c[j]}/Hurst_error_${i}_$j


mv fort.117 $add/${c[i]}_${c[j]}/gamma_$i
mv fort.1177 $add/${c[i]}_${c[j]}/gamma_error_$i

mv fort.118 $add/${c[i]}_${c[j]}/gamma_$j
mv fort.1188 $add/${c[i]}_${c[j]}/gamma_error_$j

mv fort.119 $add/${c[i]}_${c[j]}/gamma_${i}_$j
mv fort.1199 $add/${c[i]}_${c[j]}/gamma_error_${i}_$j

mv fort.1114 $add/${c[i]}_${c[j]}/beta_first
mv fort.1115 $add/${c[i]}_${c[j]}/beta_second

#Plot
cd ${c[i]}_${c[j]}/

#h(q)
gnuplot << eof
set terminal postscript eps color enhanced "Helvetica" 20
#set grid
set title "${c[i]}-${c[j]}" font "Arial, 30"

set lmargin at screen 0.07
set rmargin at screen 1.75
set bmargin at screen 0.07
set tmargin at screen 1.65

set xlabel offset 0,-1.5 "q" font "Times-Roman, 35"
set ylabel offset -1,0 "h" font "Times-Roman, 35"
set xtics offset 0,-1 font "Times-Roman, 27"
set ytics offset -1,0 font "Times-Roman, 27"
set size 2,2
```

```
set pointsize 2.5
set output 'h(q).eps'
plot "./hq_$i" w errorbar lt 1 lc rgb "red" lw 10 t "{/Times-
Roman=20 DFA-${c[i]}}"
replot "./hq_$j" w errorbar lt 1 lc rgb "blue" lw 10 t "{/Times-
Roman=20 DFA-${c[j]}}"
replot "./hq_${i}_$j" w errorbar lt 1 lc rgb "green" lw 10 t "{/
Times-Roman=20 DCCA}"

eof

#Tau(q)
gnuplot << eof
set terminal postscript eps color enhanced "Helvetica" 20
#set grid
set title "${c[i]}-${c[j]}" font "Arial, 30"

set lmargin at screen 0.07
set rmargin at screen 1.75
set bmargin at screen 0.07
set tmargin at screen 1.65

set xlabel offset 0,-1.5 "q" font "Times-Roman, 35"
set ylabel offset -1,0 "{/Symbol t}" font "Times-Roman, 35"
set xtics offset 0,-1 font "Times-Roman, 27"
set ytics offset -1,0 font "Times-Roman, 27"
set size 2,2

set pointsize 2.5
set output 'tau(q).eps'
plot "./tauq_$i" w errorbar lt 1 lc rgb "red" lw 10 t "{/Times-
Roman=20 DFA-${c[i]}}"
replot "./tauq_$j" w errorbar lt 1 lc rgb "blue" lw 10 t "{/Times-
Roman=20 DFA-${c[j]}}"
replot "./tauq_${i}_$j" w errorbar lt 1 lc rgb "green" lw 10 t "{/
Times-Roman=20 DCCA}"

eof

#D(q)
gnuplot << eof
set terminal postscript eps color enhanced "Helvetica" 20
#set grid
set title "${c[i]}-${c[j]}" font "Arial, 30"

set lmargin at screen 0.07
set rmargin at screen 1.75
set bmargin at screen 0.07
set tmargin at screen 1.65

set xlabel offset 0,-1.5 "q" font "Arial, 35"
set ylabel offset -1,0 "D" font "Arial, 35"
set xtics offset 0,-1 font "Times-Roman, 27"
```

```
set ytics offset -1,0 font "Times-Roman, 27"
set size 2,2

set pointsize 2.5
set output 'D(q).eps'
plot "./Dq_$i" w errorbar lt 1 lc rgb "red" lw 10 t "{/Times-
Roman=20 DFA-${c[i]}}"
replot "./Dq_$j" w errorbar lt 1 lc rgb "blue" lw 10 t "{/Times-
Roman=20 DFA-${c[j]}}"
replot "./Dq_${i}_$j" w errorbar lt 1 lc rgb "green" lw 10 t "{/
Times-Roman=20 DCCA}"

eof

#F(alpha)
gnuplot << eof
set terminal postscript eps color enhanced "Helvetica" 20
#set grid
set title "${c[i]}-${c[j]}" font "Arial, 30"

set lmargin at screen 0.07
set rmargin at screen 1.75
set bmargin at screen 0.07
set tmargin at screen 1.65

set yrange [0:1.2]

set xlabel offset 0,-1.5 "{/Symbol a}" font "Arial, 35"
set ylabel offset -1,0 "f" font "Times-Roman, 40"
set xtics offset 0,-1 font "Times-Roman, 27"
set ytics offset -1,0 font "Times-Roman, 27"
set size 2,2

set pointsize 2.5
set output 'F(a).eps'
plot "./falpha_$i" w errorbar lt 1 lc rgb "red" lw 10 t "{/Times-
Roman=20 DFA-${c[i]}}"
replot "./falpha_$j" w errorbar lt 1 lc rgb "blue" lw 10 t "{/Times-
Roman=20 DFA-${c[j]}}"
replot "./falpha_${i}_$j" w errorbar lt 1 lc rgb "green" lw 10 t "{/
Times-Roman=20 DCCA}"

eof

cd ..

echo ${c[i]}_${c[j]}
done
done

rm -f dcca
rm -f dfa_r.mod
rm -f fs_DCCA
rm -f fs_first
```

```
rm -f fs_second
rm -f PDF_first
rm -f PDF_second
./plot.sh
```

بسم الله الرحمن الرحیم

SBU-Cluster
SARMAD

آشنایی با سامانه رایانش سریع دانشگاه شهید بهشتی

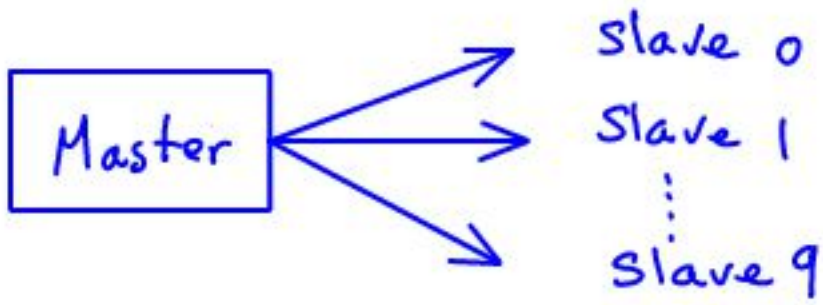
ارائه دهنده: آقای مهندس آرمین احمدزاده

در این متن خلاصه‌ای جهت آشنایی با خوشه محاسباتی دانشگاه شهید بهشتی آمده است. مطالب تکمیلی در سایت معاونت پژوهشی دانشگاه قابل دریافت است.

فهرست

۱- ویژگی های خوشه محاسباتی

این خوشه محاسباتی در حال حاضر دارای یک Master و تعداد ۱۰ ← ۱۰ Node است



هرکدام از Slave ها دارای مشخصات زیر است

64 Cores

10T flops

128 GB RAM و GPU-Node : 2688
8GB RAM

IP این Master عبارت است از :

192.168.220.100

- در حال حاضر هر کاربر حجمی برابر با 5 GB در اختیار خواهد داشت
- سیستم عامل این خوشه Scientific linux 6.5 است
- همچنین برخی از نرم افزارهای کاربردی در حال حاضر دارد. در صورت نیاز به نرم افزاری دیگر با تماس با مسئول سیستم امکان نصب در صورت وجود نرم افزار فراهم است

۲- نحوه اتصال

۲-۱ : از طریق Windows

با یک نرم افزار رایگان PuTTy می توان به این مجموعه متصل شد.
در بخش مربوطه IP و سپس Port No : 22 تنظیم می شود
سپس ID و گذرواژه داده می شود

۲-۲ از طریق linux

با یک دستور

ssh userID @ 192.168.220.100 ↵

Pass : ↵

وارد فضای خوشه محاسباتی خواهیم شد.

۳- مدیریت و اجرای برنامه

درحال حاضر روند اجرای برنامه براساس سیاست first in → first out است

مهم‌ترین دستورات برای اجرای یک برنامه عبارتند از:

qsub	submit a job
qstat	Monitor the status of jobs
qdel	Terminate a job

← برای اجرای یک برنامه حتماً باید یک script نوشت که نوع فایل sh. یا csh. است

فرض کنید یک اجراگر آماده شده بنام sadeg.sh برای اجرای آن

qsub sadegh.sh ↵

Job ID نمایش داده می‌شود که بسیار مهم است

← برای بررسی وضعیت برنامه ارسال شده جهت اجرا از دستور زیر استفاده می‌کنیم

qstat ↵

Job ID	Q	R	E	C
	(Queued)	(Running)	(Error)	(Complete)
	در صف	در حال اجرا	خطا	اتمام

اگر خطایی رخ داده باشد درهمان جایی که فایل اجرایی است یک فایل درست می‌شود و شرح خطا در آن می‌آید

نام فایل را خودمان در فایل script تعیین خواهیم کرد.

نکته: گاهی مواقع وضعیت برنامه R ات وی واقعاً اجرایی دلکاریت برای اینکه اطمینان حاصل کنیم
از صحت برنامه ابتدا در کامپیوتر خود از صحت برنامه اطمینان حاصل کنیم بعد برای اجرا بفرستیم.

← برای بررسی وضعیت برنامه ها دستورات اضافی (flag) به شرح زیر وجود دارد.

qstat -a	shows all jobs
qstat -r	shows all running jobs
qstat -f	shows Detail of jobs
qstat -n	shows nodes running

← برای حذف برنامه از دستور زیر استفاده می کنیم:

qdel <Job ID> ← شماره مربوط به برنامه درج شود

نکته گاهی مواقع با یک دستور، بالا نمی توان برنامه را متوقف ساخت در این حالت با ارسال ID مربوط به برنامه، سرپرست خوشه از طریق ایمیل درخواست خود را مطلع خواهیم کرد.

۴- script file

همانطوریکه گفته شد باید یک فایل ساخت که در آن مشخص شود چه کارهایی قرار است انجام گیرد ساختار کلی این فایل به قرار زیر است:

* PBS -S	نوع فایل (دلخواه)
* PBS -N	<Job Name> ← کاربر تعیین می کند
* PBS -I	nodes=<* Nodes: Defult=1>: ← حداکثر ۱۵
	ppn=<* Cores in each node e.g.=24> ← حداکثر ۶۴ است

*PBS -q batch ← تعداد صف هایی که دقیقاً همین اسم را دارند تغییر

*PBS -o $HOME/<job folder>/<job out put file name>

برای مثال: $PBS_JOBID.out

شماره برنامه را اینجا قرار می دهد

*PBS -e $HOME/<job folder>/<job error file>

برای مثال: $PBS_JOBID.error

*PBS -l Wall time = 12:00:00 ← تخمینی از مدت زمان اجرای برنامه

NP= Node x Core در صورتی که از MPI استفاده نشود نیاز نیست

{ بقیه قسمت های برنامه که در فایل Bash هستند }

- توجه عبارت ./a.out < Address >

برنامه a.out را اجرا و خروجی آن را در آدرس داده شده می ریزد

- برنامه هایی همچون Vim و ... emacs برای ویرایش فایل script استفاده می شوند

۵- انتقال نتایج

۵-۱ Windows

اگر در سیستم عامل ویندوز هستید بهتر است نرم افزار رایگان Winscp استفاده کرد

با نصب این برنامه و با تعیین IP و ID و Pass می‌توان به راحتی اطلاعات را استقل کرد

۵-۲ Linux

برای گرفتن نتایج در لینوکس از دستور زیر استفاده می‌کنیم یعنی:

scp User Name @ 192.168.220.100 :/ < file Address on Cluster > ~ < Address in your Machine >

space

برای ارسال فایل‌های مورد نیاز ماشین خود به خوشه از دستور زیر استفاده می‌کنیم

scp < file Address and file name > ~ User Name @ 192.168.220.100 :/ < Address >

space

نکته پایانی اینکه در هر User Name مجموعه‌ای از Documents و اطلاعیه‌ها به طور مداوم قرار می‌گیرد

برای دریافت اطلاعات اضافی در خصوص نوشتن فایل و دستورات Bash در linux

آدرس زیر را مراجعه نمایید:

www.smouahed.ir
بخش Courses بخش Computational physics
بخش Some necessary things for Programming

تهیه کننده: سید مهرصادق برحد ۹۳/۵/۱۲

```
#! /bin/bash

for ch in gamma Hurst delta_falpha ; do  # Hurst delta_falpha
#for ch in gamma Hurst  ; do  # Hurst delta_falpha


#Reading of contries list
i=0
for name in `cat list_arrange` ; do
  let "i = $i+1"
  c[i]=$name
#echo $name
done

rm -f ${ch}_cross
rm -f $ch
rm -f ${ch}_error_cross
rm -f ${ch}_error

let "num = $i"

let "num1 = $num-1"

for ((i=1; i<=$num1; i++)); do
let "k = $i+1"

cd ${c[i]}_${c[k]}/

a=$(cat ${ch}_${i})
b=$(cat ${ch}_error_${i})
#    echo "$j $c" >> gamma
#    echo "$j $d" >> Hurst
if [ "$k" = "$num" ]; then
    c=$(cat ${ch}_${k})
    d=$(cat ${ch}_error_${k})
fi
cd ..

echo "$i  $a" >> $ch
echo "$i  $b" >> ${ch}_error
if [ "$k" = "$num" ]; then
    echo "$k  $c" >> ${ch}
    echo "$k  $d" >> ${ch}_error
fi
done

#for ((j=$k; j<=$k; j++)); do
#done

for ((i=1; i<=$num; i++)); do
let "k = $i+1"
for ((j=$k; j<=$num; j++)); do

cd ${c[i]}_${c[j]}/
```

```
a=$(cat ${ch}_${i}_$j)
b=$(cat ${ch}_error_${i}_$j)
#a=$(cat gamma_DCCA)
#a=$(echo $a | awk '{ print $1 }')
#b=$(cat Hurst_DCCA)
#b=$(echo $b | awk '{ print $1 }')
#printf "%s" "$a   " >> gamma_cross
#printf "%s" "$b   " >> Hurst_cross

cd ..

echo $i  $j  $a >> ${ch}_cross
echo $i  $j  $b >> ${ch}_error_cross

done

done

ifort Merger.f90  -o Merger

./Merger<< eof
$num
${ch}
eof

rm -f Merger

#gamma for countries
gnuplot << eof
set terminal postscript eps color enhanced "Helvetica" 20

#set title "${c[i]}-${c[j]}" font "Arial, 30"
#set lmargin at screen 0.07
#set rmargin at screen 1.75
#set bmargin at screen 0.07
#set tmargin at screen 1.65
#set yrange [0:1.2]

set xlabel offset 0,-1.5 "Countries" font "Arial, 35"
set ylabel offset -1,0 "${ch}" font "Times-Roman, 40"

#set xtics ()
set xtics offset 0,-1 font "Times-Roman, 27"
set ytics offset -1,0 font "Times-Roman, 27"
set size 2,2
set grid
set pointsize 2.5
set output '${ch}.eps'
plot "./${ch}_final" w errorbar lt 1 lc rgb "red" lw 10 t "{/Times-
Roman=20 }"

eof
```

```
gnuplot << eof
set terminal postscript eps color enhanced "Helvetica" 20

set title "${ch}  exponent for countries" font "Arial, 30"

#set lmargin at screen 0.07
#set rmargin at screen 1.75
set bmargin at screen 0.12
#set tmargin at screen 1.65
#set yrange [0:1.2]
#set xtics 5 out nomirror
#set ytics 5 out nomirror
#set xlabel offset 0,-1.5 "Countries" font "Arial, 35"
#set ylabel offset -1,0 "{/Symbol g}" font "Times-Roman, 40"


set autoscale xfix
set autoscale yfix
set xtics 1,1,48 offset -2.2,-1 font "Times-Roman, 20"
set ytics 1,1,48 offset -1,-1 font "Times-Roman, 20"
#set tics scale 0,0.001
set mxtics 2
set mytics 2
set grid front mxtics mytics lw 1.5 lt -1 lc rgb 'white

set size 2,2
#set grid
set pointsize 2.5
set output '${ch}_matrix.eps'
set palette gray negative

plot "${ch}_matrix" matrix w image noti

eof

gnuplot << eof
set terminal postscript eps color enhanced "Helvetica" 20

set title "${ch}_error  exponent for countries" font "Arial, 30"

#set lmargin at screen 0.07
#set rmargin at screen 1.75
set bmargin at screen 0.12
#set tmargin at screen 1.65
#set yrange [0:1.2]
#set xtics 5 out nomirror
#set ytics 5 out nomirror
#set xlabel offset 0,-1.5 "Countries" font "Arial, 35"
#set ylabel offset -1,0 "{/Symbol g}" font "Times-Roman, 40"


set autoscale xfix
set autoscale yfix
```

```
set xtics 1,1,48 offset -2.2,-1 font "Times-Roman, 20"
set ytics 1,1,48 offset -1,-1 font "Times-Roman, 20"
#set tics scale 0,0.001
set mxtics 2
set mytics 2
set grid front mxtics mytics lw 1.5 lt -1 lc rgb 'white

set size 2,2
#set grid
set pointsize 2.5
set output '${ch}_error_matrix.eps'
set palette gray negative

plot "${ch}_error_matrix" matrix w image noti

eof


done
```